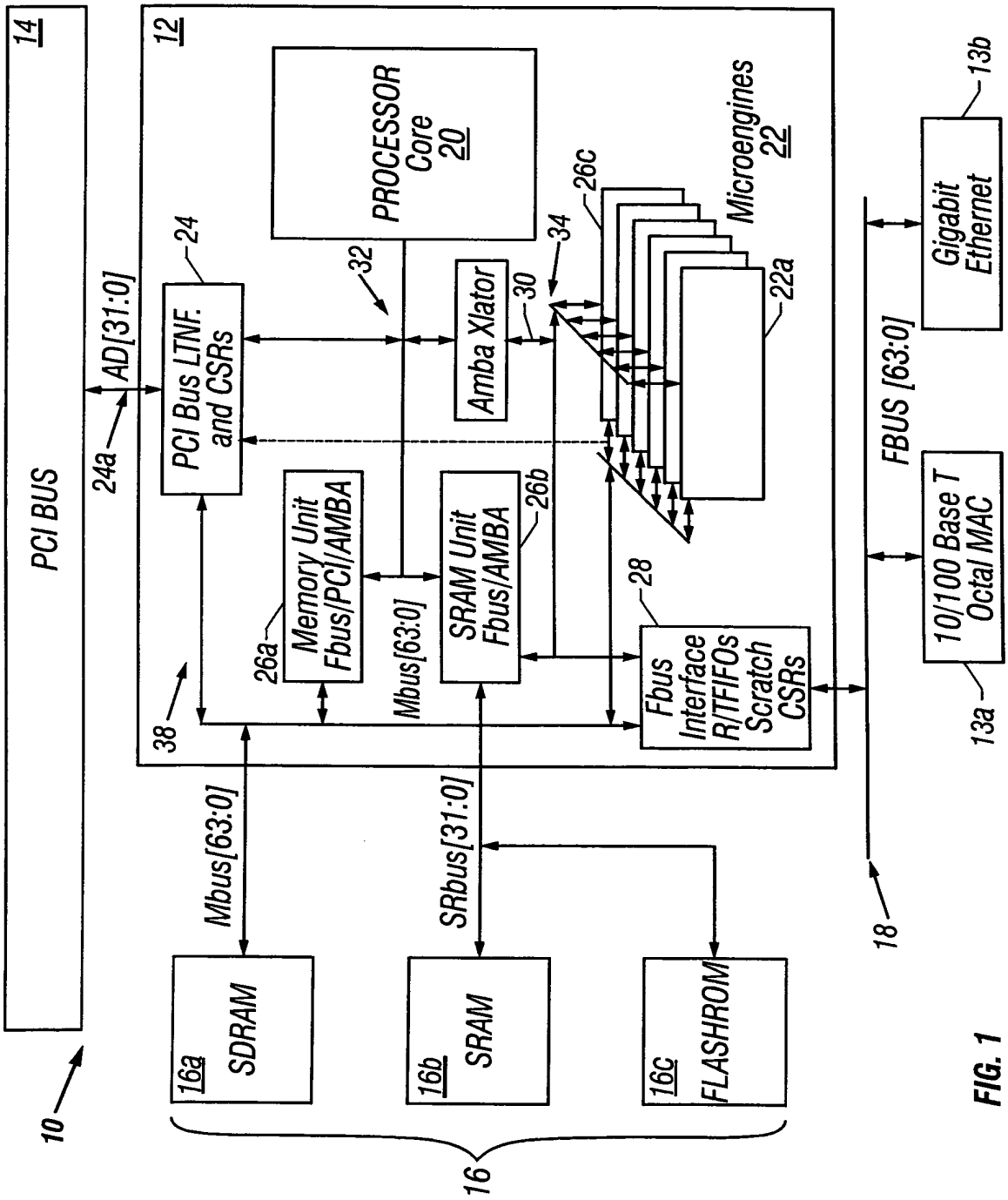

14
PCI BUS
10
12
PROCESSOR Core 20
Microengines 22
22a
26c
34
32
24
PCI Bus LTNF. and CSRs
AD[31:0]
24a
Amba Xlator
30
Memory Unit Fbus/PCI/AMBA
26a
SRAM Unit Fbus/AMBA
26b
Mbus[63:0]
28
Fbus Interface R/TFIFOs Scratch CSRs
38
Mbus[63:0]
SRbus[31:0]
FBUS [63:0]
Gigabit Ethernet
13b
10/100 Base T Octal MAC
13a
18
16a SDRAM
16b SRAM
16c FLASHROM
16

FIG. 1

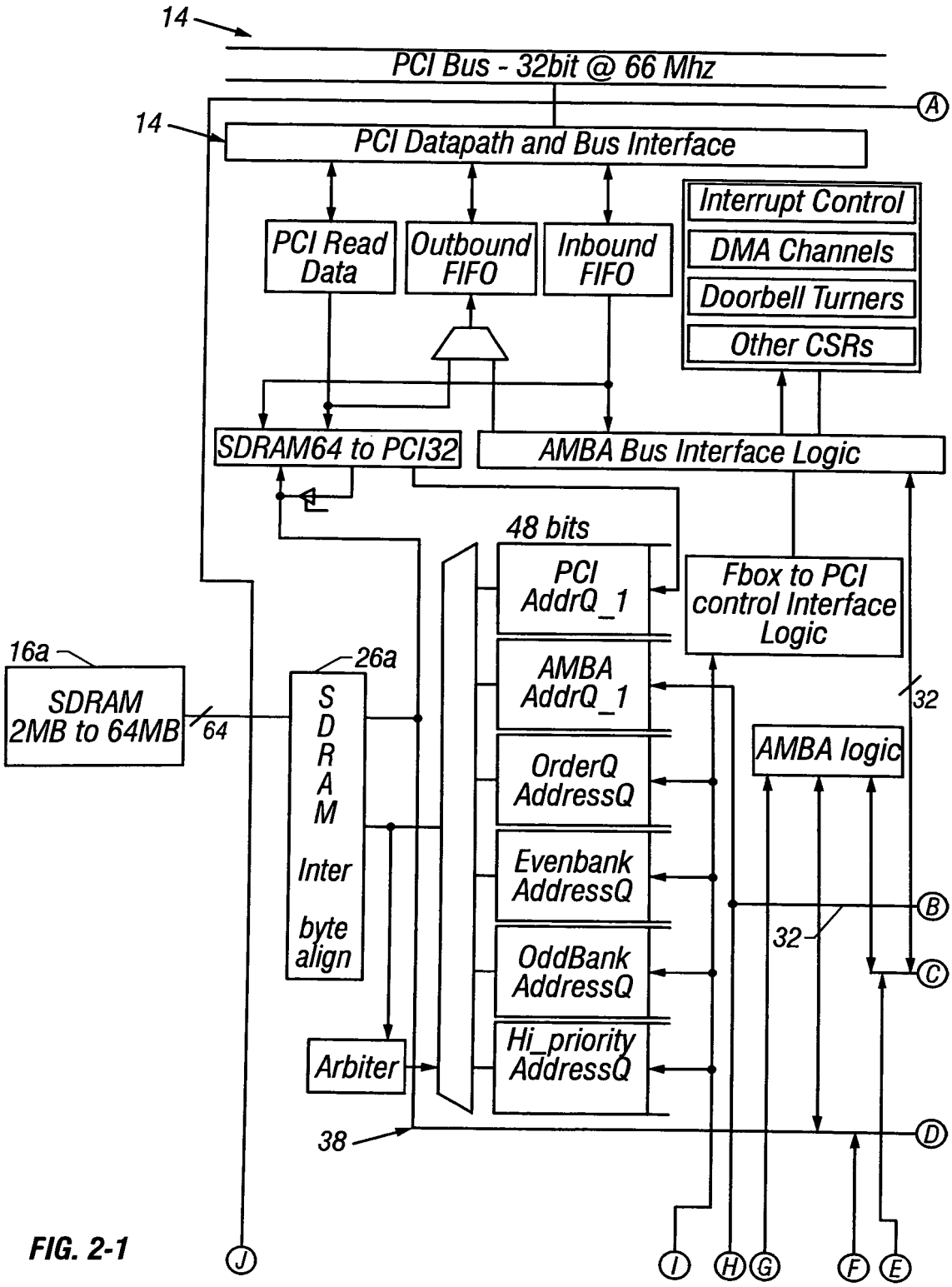

14
PCI Bus - 32bit @ 66 Mhz
A
14
PCI Datapath and Bus Interface
PCI Read Data
Outbound FIFO
Inbound FIFO
Interrupt Control
DMA Channels
Doorbell Turners
Other CSRs
SDRAM64 to PCI32
AMBA Bus Interface Logic
48 bits
PCI AddrQ_1
Fbox to PCI control Interface Logic
16a
SDRAM 2MB to 64MB
64
26a
S D R A M Inter byte align
AMBA AddrQ_1
32
AMBA logic
OrderQ AddressQ
Evenbank AddressQ
B
32
OddBank AddressQ
C
Arbiter
Hi_priority AddressQ
38
D
J
I
H
G
F
E

FIG. 2-1

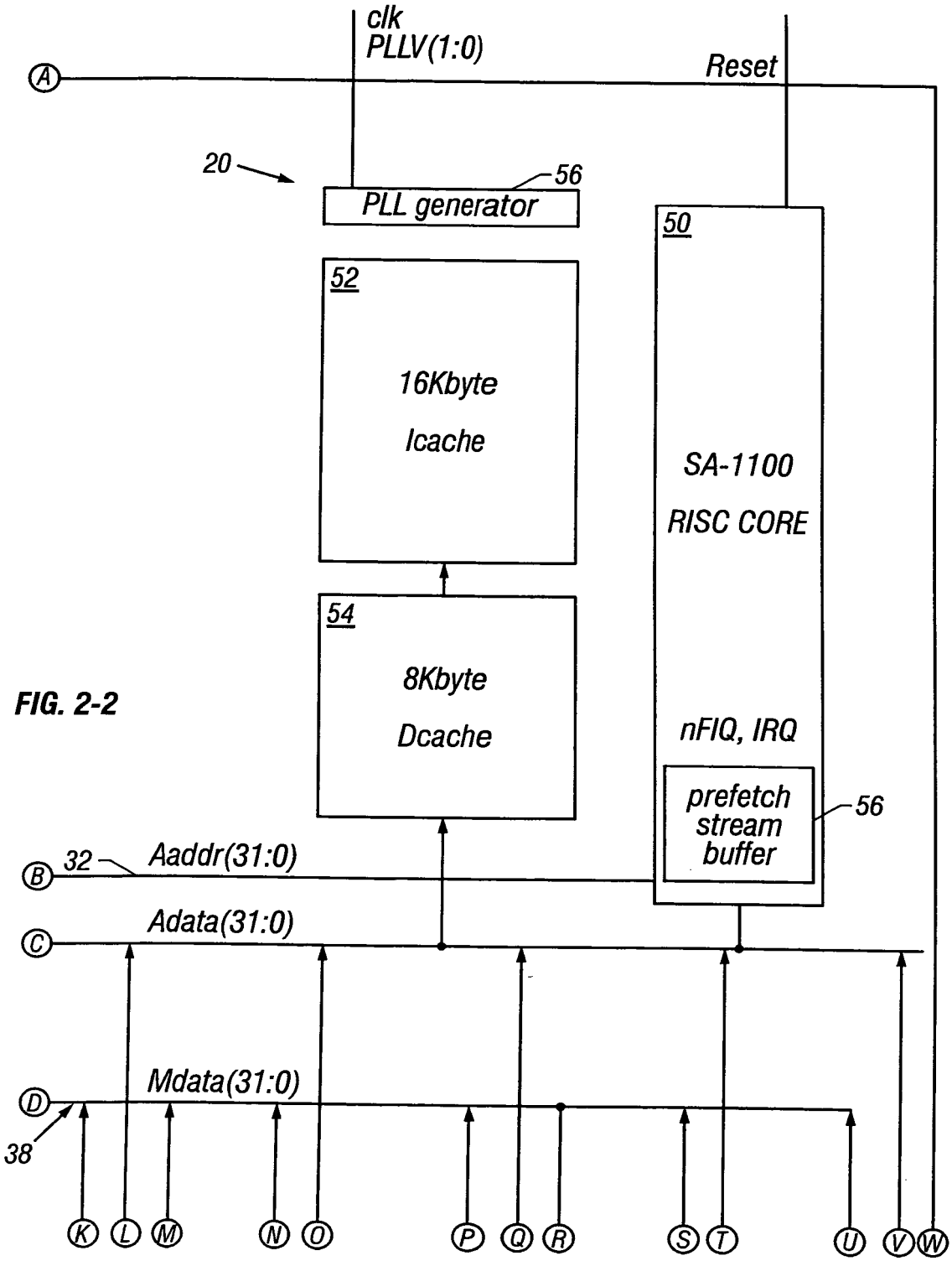

clk
PLLV(1:0)
Reset
A
20
56
PLL generator
50
52
16Kbyte
Icache
SA-1100
RISC CORE
54
8Kbyte
Dcache
nFIQ, IRQ
prefetch
stream
buffer
56
32
Aaddr(31:0)
B
Adata(31:0)
C
Mdata(31:0)
D
38
K
L
M
N
O
P
Q
R
S
T
U
V
W

FIG. 2-2

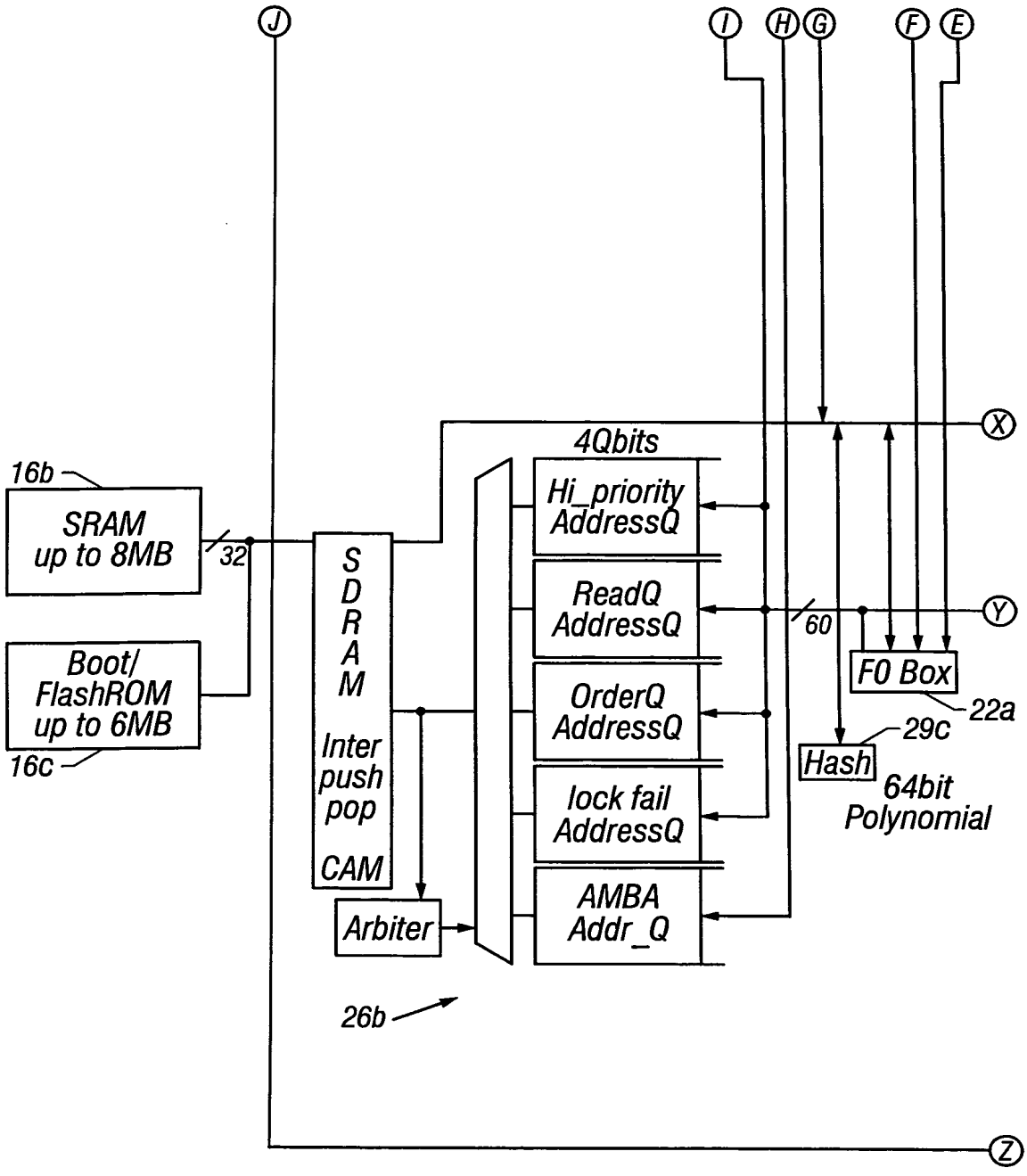

J
I
H
G
F
E
X
Y
Z
4Qbits
16b
SRAM
up to 8MB
32
Boot/
FlashROM
up to 6MB
16c
SDRAM
Inter
push
pop
CAM
Arbiter
Hi_priority
AddressQ
ReadQ
AddressQ
OrderQ
AddressQ
lock fail
AddressQ
AMBA
Addr_Q
60
F0 Box
22a
29c
Hash
64bit
Polynomial
26b

FIG. 2-3

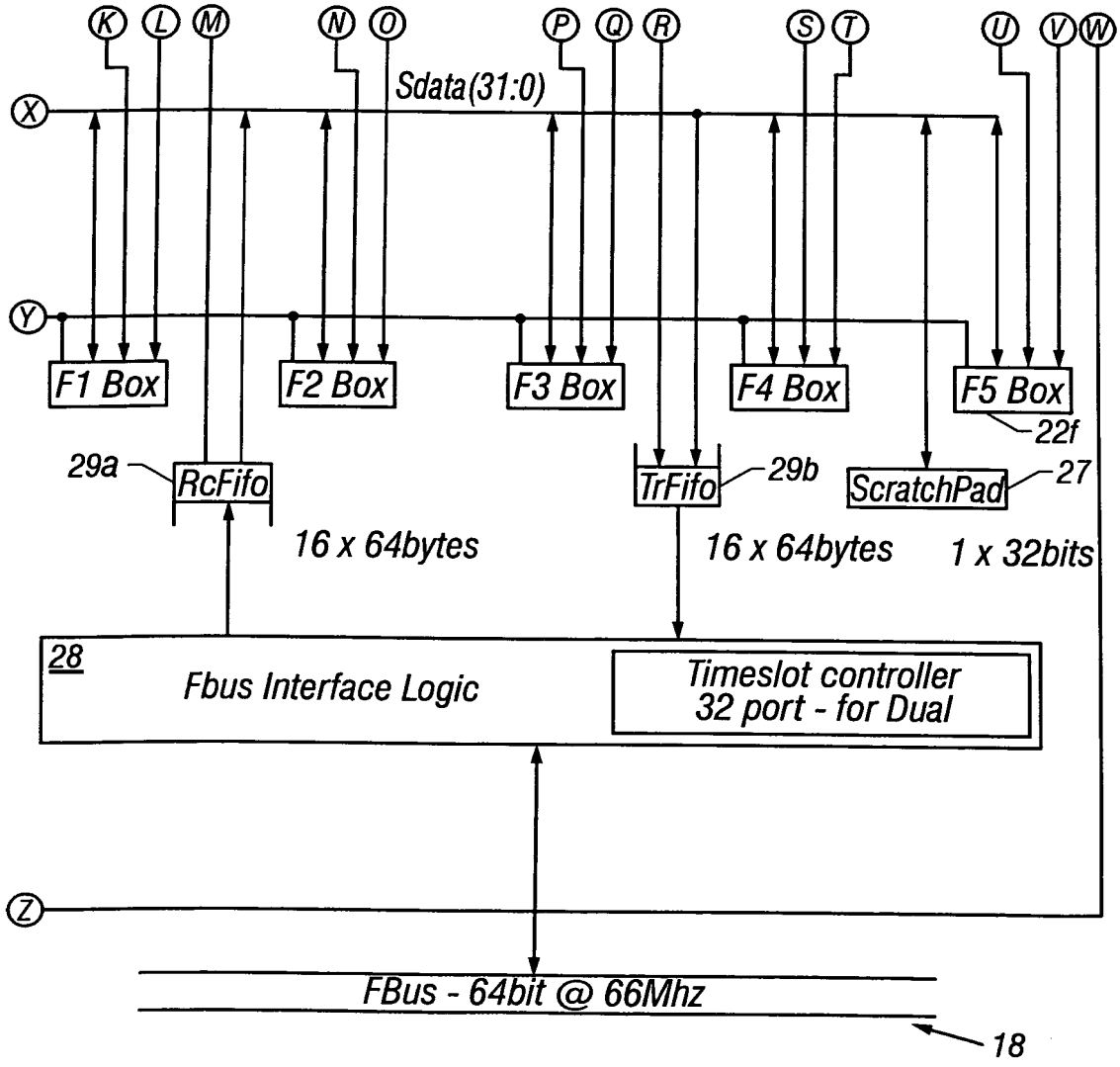

K
L
M
N
O
P
Q
R
S
T
U
V
W
Sdata(31:0)
X
Y
F1 Box
F2 Box
F3 Box
F4 Box
F5 Box
22f
29a
RcFifo
16 x 64bytes
TrFifo
29b
16 x 64bytes
ScratchPad
27
1 x 32bits
28
Fbus Interface Logic
Timeslot controller
32 port - for Dual
Z
FBus - 64bit @ 66Mhz
18

FIG. 2-4

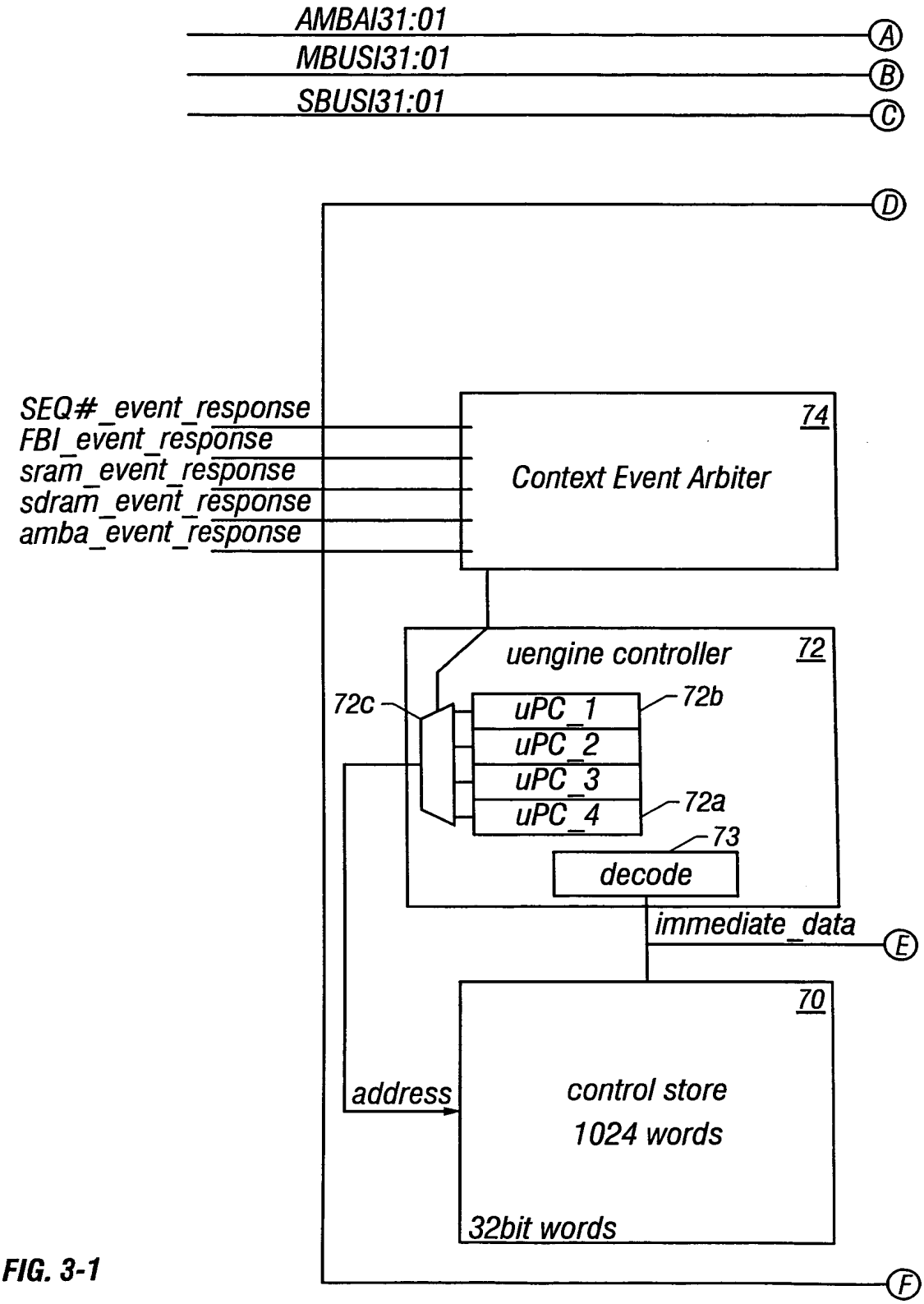

AMBAI31:01
MBUSI31:01
SBUSI31:01
A
B
C
D
SEQ#_event_response
FBI_event_response
sram_event_response
sdram_event_response
amba_event_response
74
Context Event Arbiter
uengine controller
72
72c
uPC_1
uPC_2
uPC_3
uPC_4
72b
72a
73
decode
immediate_data
E
70
address
control store
1024 words
32bit words
F

FIG. 3-1

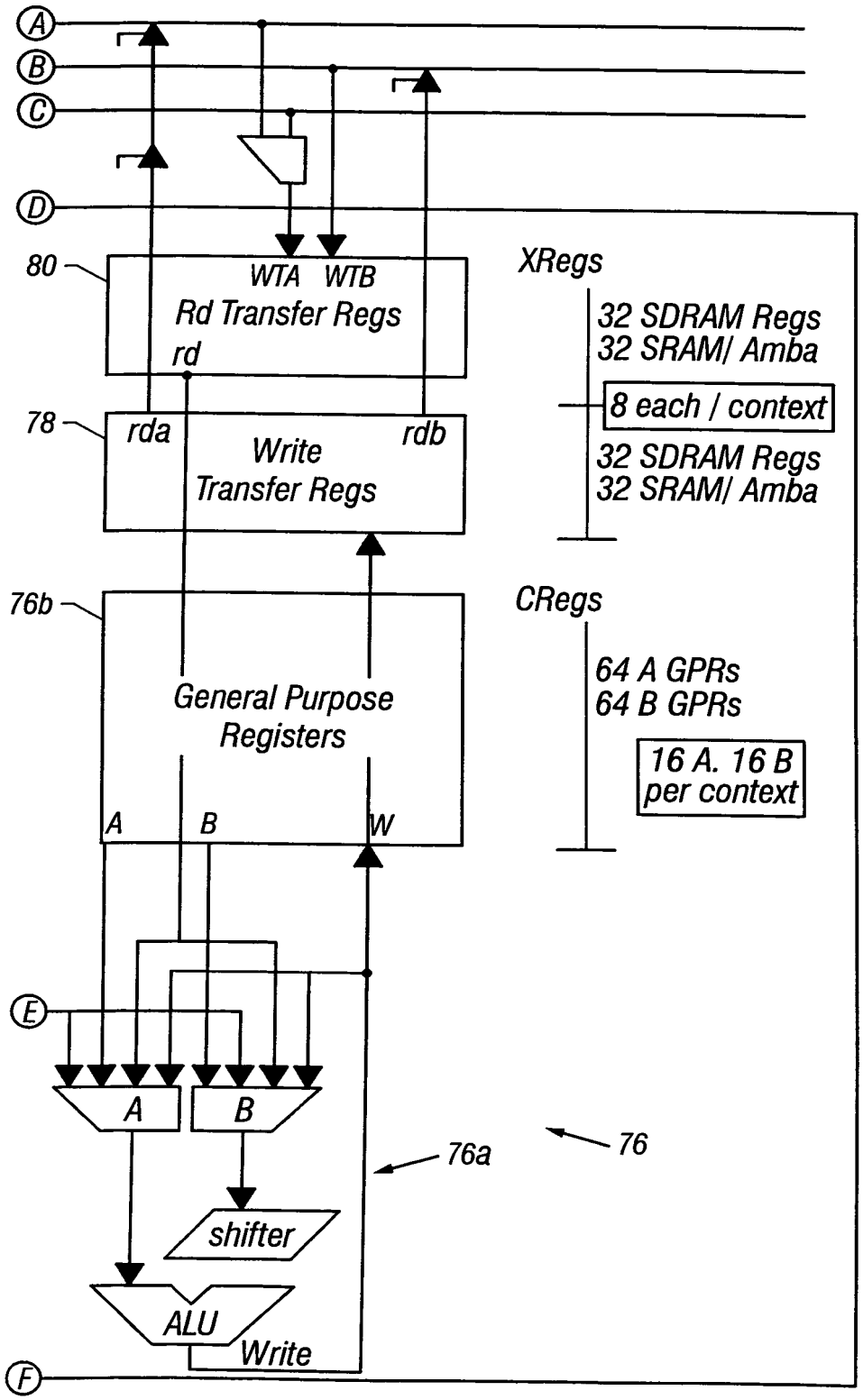

A
B
C
D
80
WTA WTB
Rd Transfer Regs
rd
78
rda
rdb
Write
Transfer Regs
76b
General Purpose
Registers
A
B
W
E
A
B
shifter
ALU
Write
F
76a
76
XRegs
32 SDRAM Regs
32 SRAM/ Amba
8 each / context
32 SDRAM Regs
32 SRAM/ Amba
CRegs
64 A GPRs
64 B GPRs
16 A. 16 B
per context

FIG. 3-2

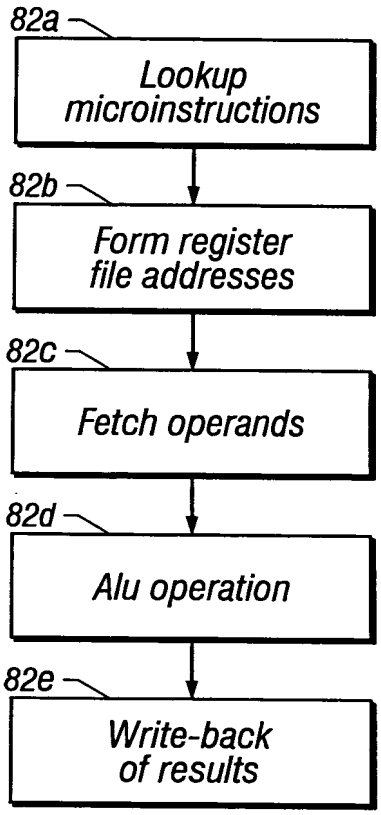
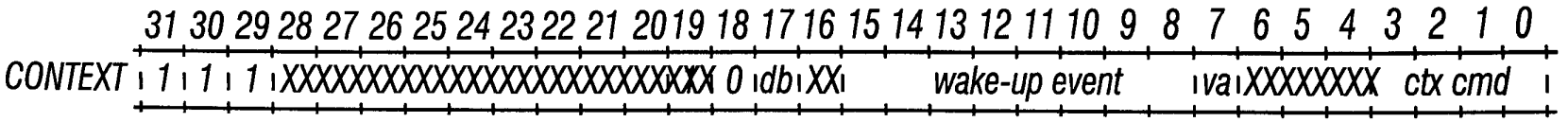

82a
Lookup microinstructions
82b
Form register file addresses
82c
Fetch operands
82d
Alu operation
82e
Write-back of results

FIG. 3A



Context Descriptors:

1) Wake-up Events (Bits 8-15)

0 = kill

1 =voluntary

2 = SRAM

4 = SDRAM

8 = FBI

16 = INTER_THREAD

32 = PCI_DMA_1

64 = PCI_DMA_2

128= SEQ_NUM_LSB

2) db -> branch defer amount (Bit 17)

3) va -> value of sequence number (Bit 7)

4) OPCODE Bits (29-31)

5) cxt_cmd

FIG. 3B

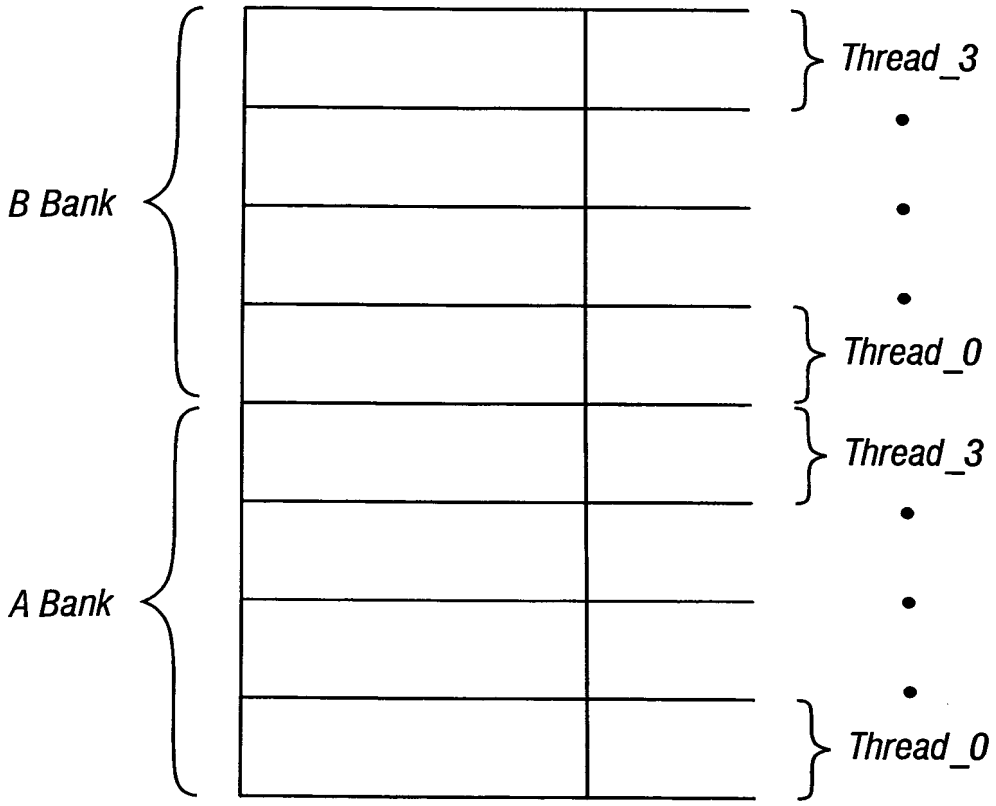


FIG. 3C

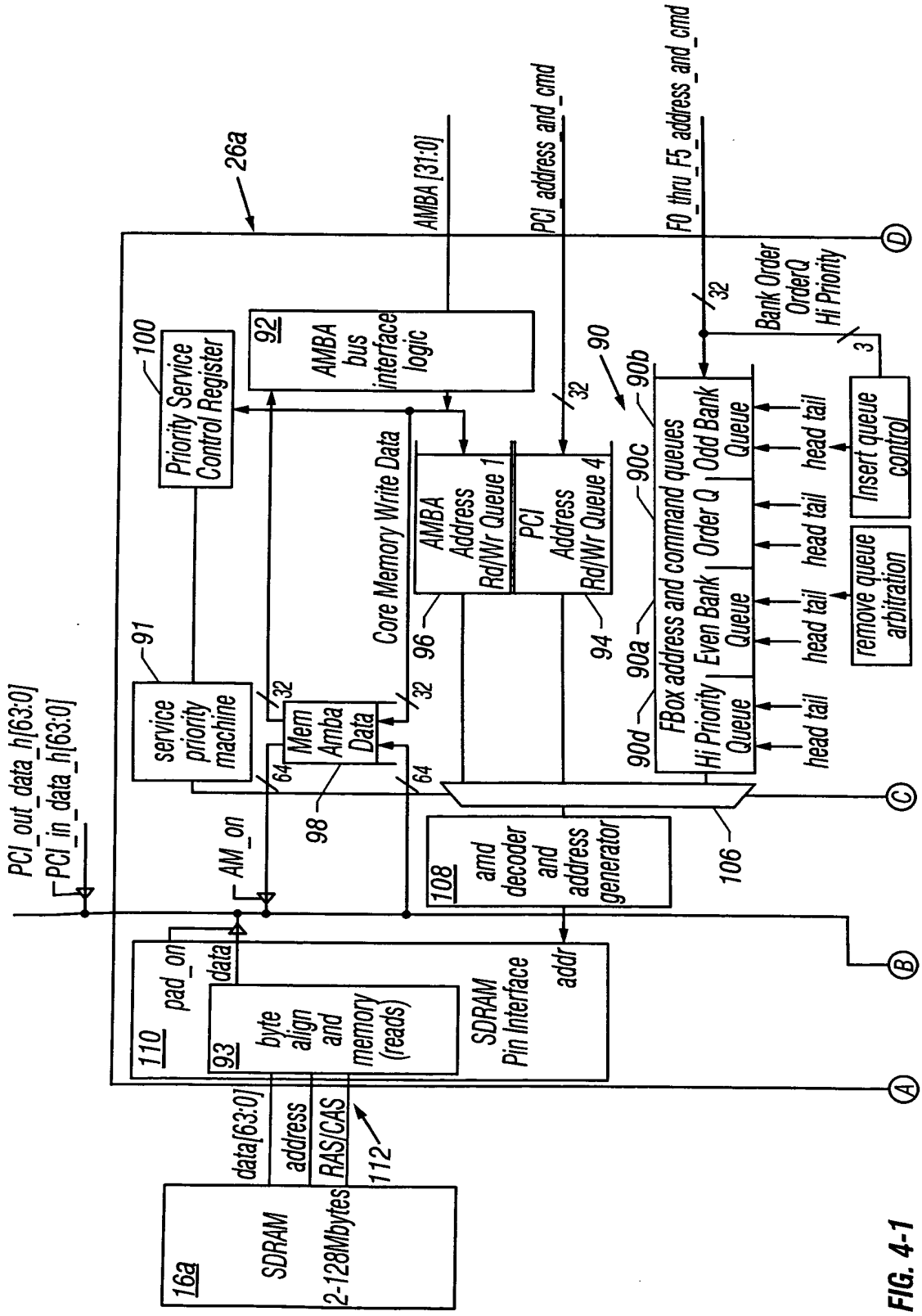

26a
AMBA [31:0]
PCI address and cmd
F0 thru F5 address and cmd
Bank Order
OrderQ
Hi Priority
32
3
92
AMBA bus interface logic
100
Priority Service Control Register
90
90b
90c
Odd Bank Queue
Order Q
Even Bank Queue
Hi Priority Queue
FBox address and command queues
head tail
Insert queue control
remove queue arbitration
Core Memory Write Data
AMBA Address Rd/Wr Queue 1
PCI Address Rd/Wr Queue 4
96
94
90a
90d
91
service priority machine
Mem Amba Data
64
98
PCI_out_data_h[63:0]
PCI_in_data_h[63:0]
AM_on
108
amd decoder and address generator
106
110
pad_on
data
93
byte align and memory (reads)
SDRAM Pin Interface
addr
data[63:0]
address
RAS/CAS
112
16a
SDRAM
2-128Mbytes
A
B
C
D

FIG. 4-1

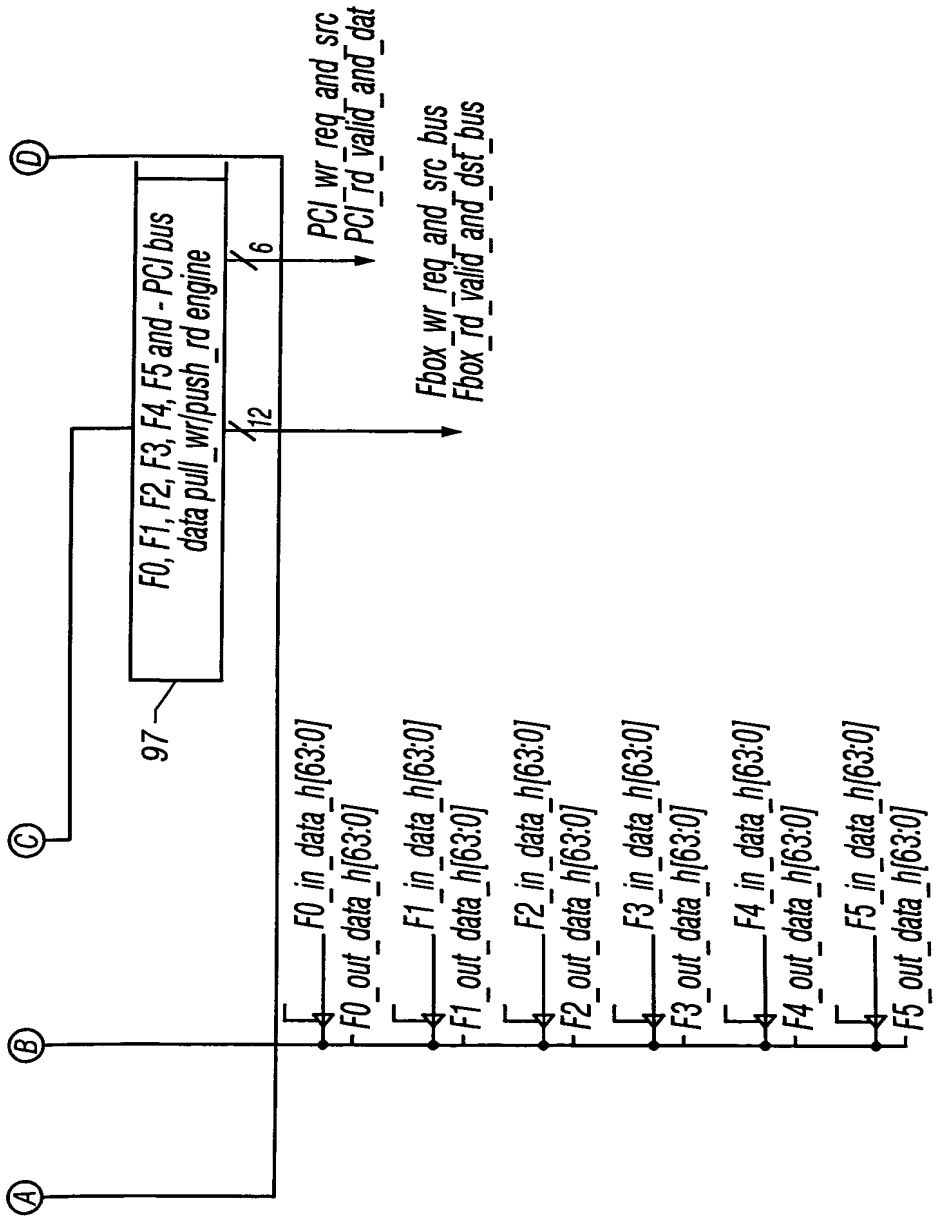

D
C
B
A
F0, F1, F2, F3, F4, F5 and - PCI bus
data pull_wr/push_rd engine
97
6
12
PCI wr_req_and_src
PCI_rd_valid_and_dat
Fbox_wr_req_and_src_bus
Fbox_rd_valid_and_dst_bus
F0_in_data_h[63:0]
F0_out_data_h[63:0]
F1_in_data_h[63:0]
F1_out_data_h[63:0]
F2_in_data_h[63:0]
F2_out_data_h[63:0]
F3_in_data_h[63:0]
F3_out_data_h[63:0]
F4_in_data_h[63:0]
F4_out_data_h[63:0]
F5_in_data_h[63:0]
F5_out_data_h[63:0]

FIG. 4-2

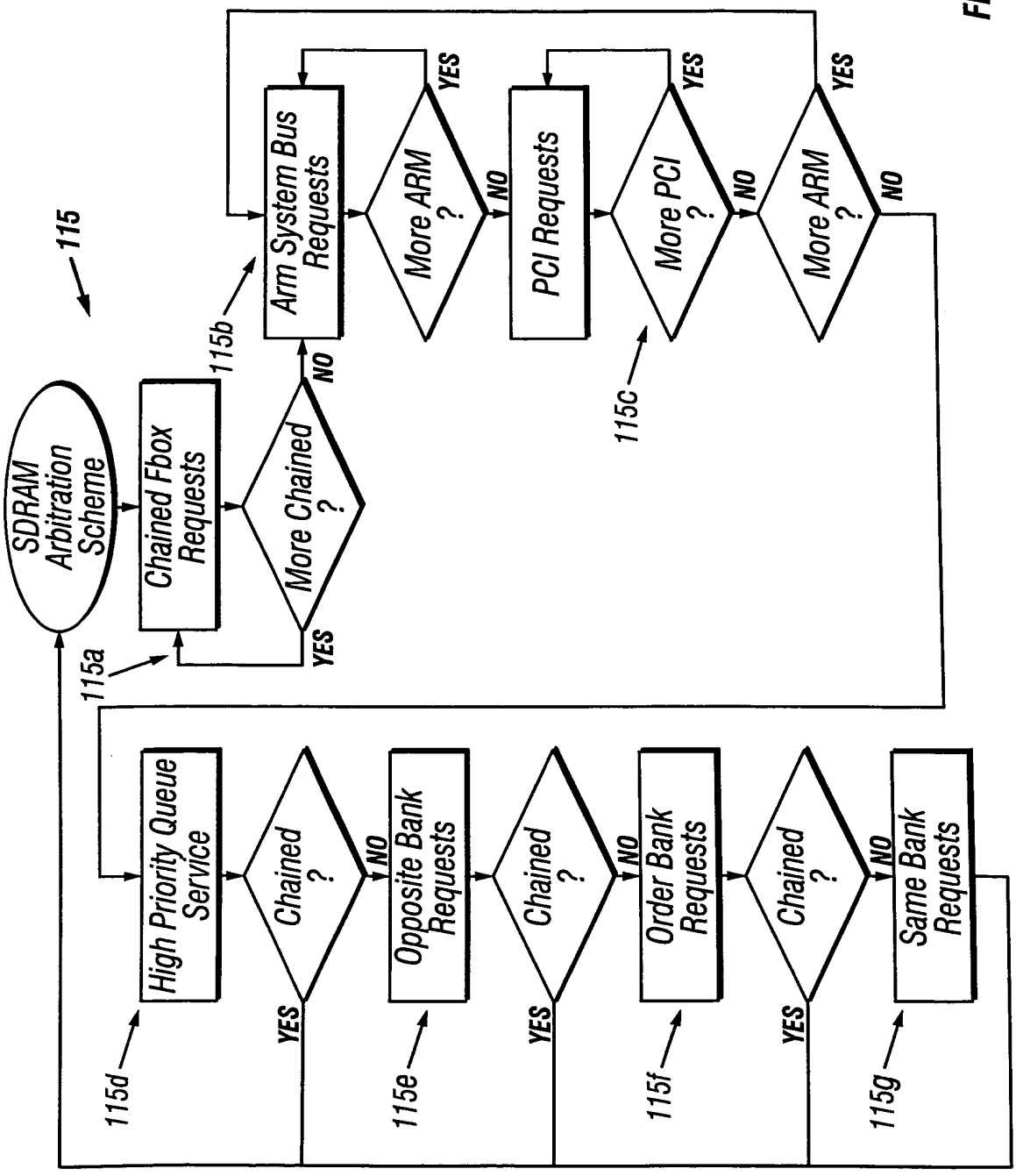

115
SDRAM Arbitration Scheme
115a
Chained Fbox Requests
More Chained ?
YES
NO
115b
Arm System Bus Requests
More ARM ?
YES
NO
PCI Requests
More PCI ?
YES
NO
115c
More ARM ?
YES
NO
High Priority Queue Service
115d
Chained ?
YES
NO
Opposite Bank Requests
115e
Chained ?
YES
NO
Order Bank Requests
115f
Chained ?
YES
NO
Same Bank Requests
115g

FIG. 4A

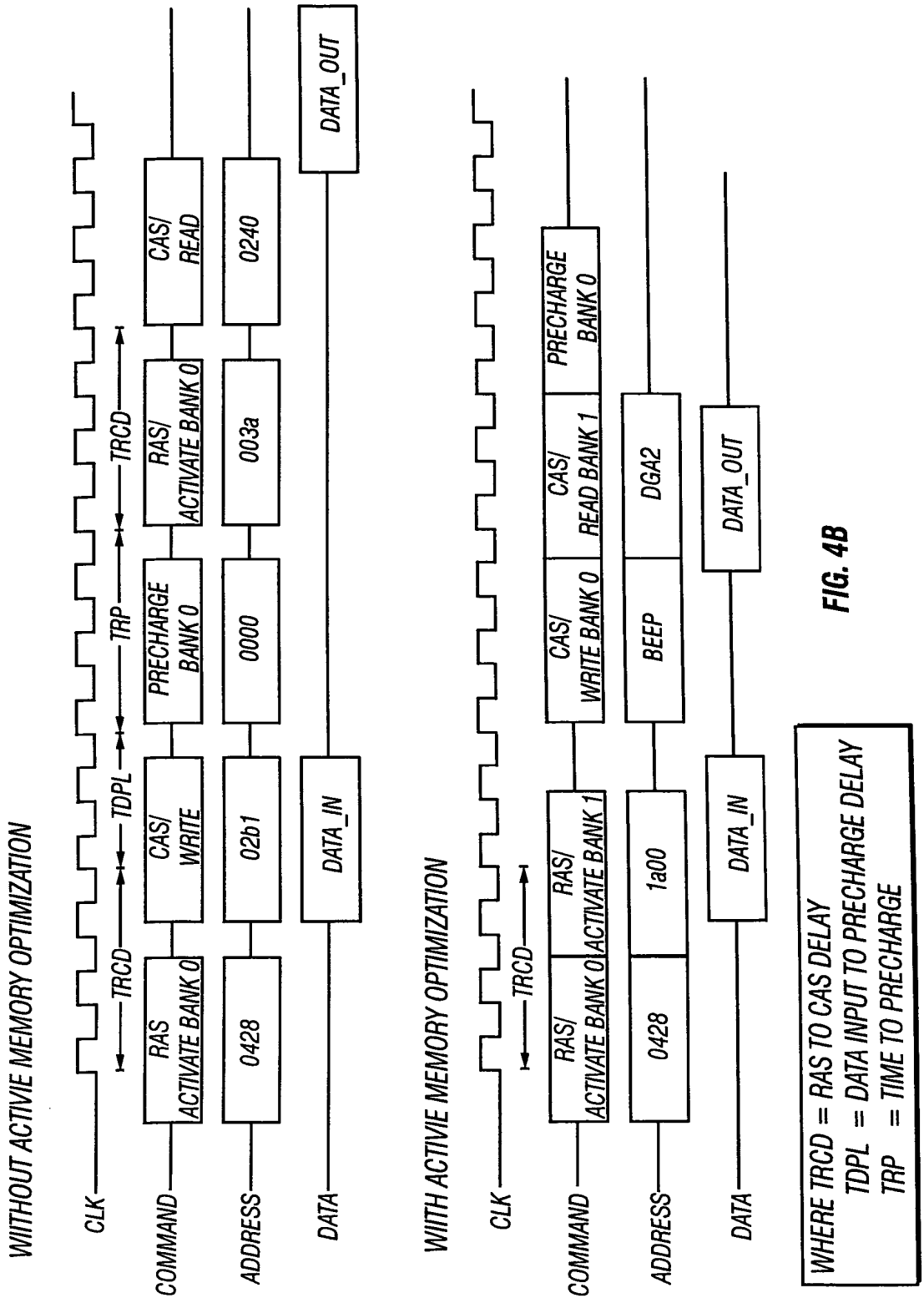

WITHOUT ACTIVIE MEMORY OPTIMIZATION
CLK
COMMAND
ADDRESS
DATA
TRCD
TDPL
TRP
TRCD
RAS ACTIVATE BANK 0
CAS/ WRITE
PRECHARGE BANK 0
RAS/ ACTIVATE BANK 0
CAS/ READ
0428
02b1
0000
003a
0240
DATA_IN
DATA_OUT
WITH ACTIVIE MEMORY OPTIMIZATION
CLK
COMMAND
ADDRESS
DATA
TRCD
RAS/ ACTIVATE BANK 0
RAS/ ACTIVATE BANK 1
CAS/ WRITE BANK 0
CAS/ READ BANK 1
PRECHARGE BANK 0
0428
1a00
BEEP
DGA2
DATA_IN
DATA_OUT
WHERE TRCD = RAS TO CAS DELAY
TDPL = DATA INPUT TO PRECHARGE DELAY
TRP = TIME TO PRECHARGE

FIG. 4B

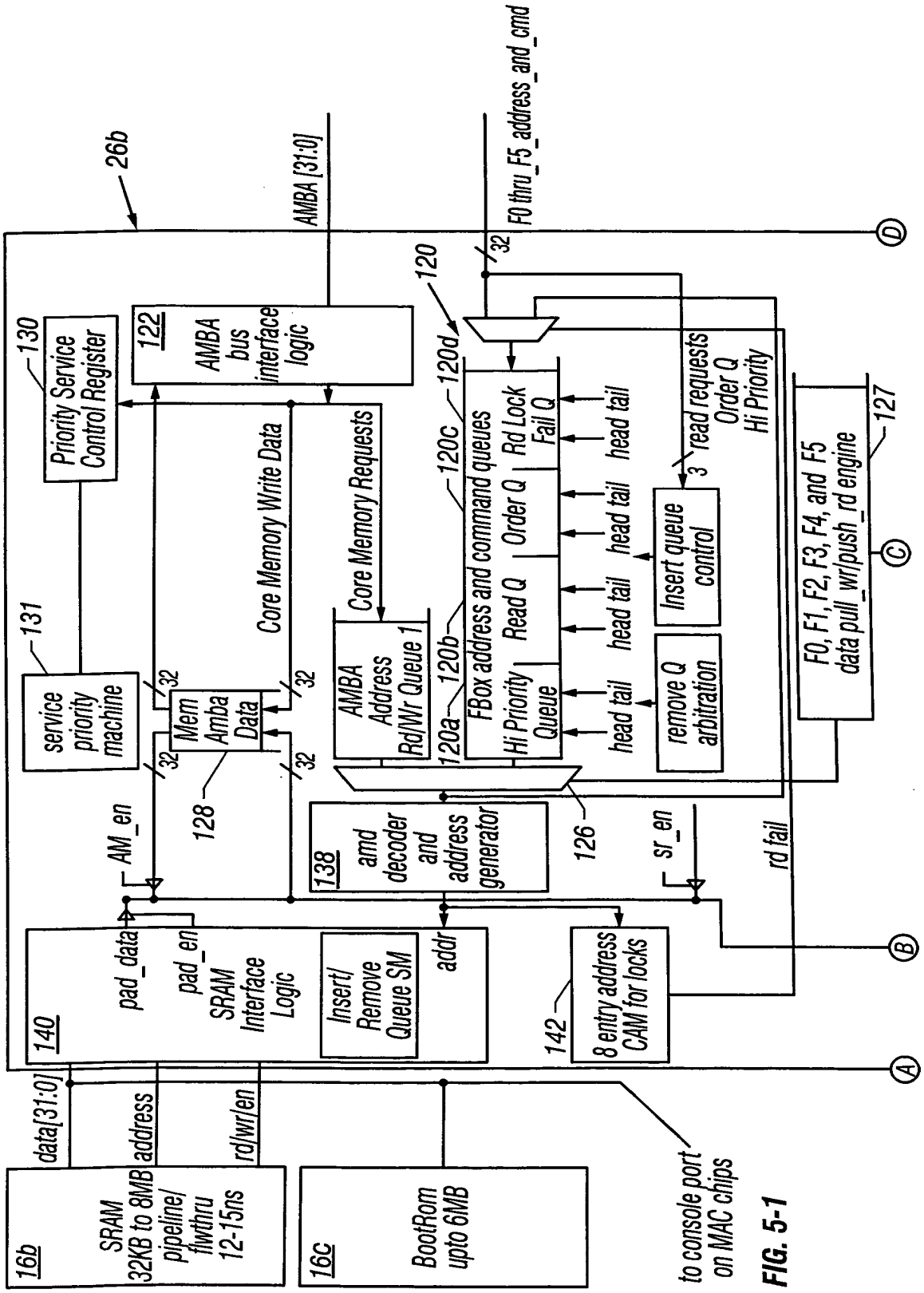

26b
AMBA [31:0]
F0 thru F5 address and cmd
32
120
130
Priority Service Control Register
122
AMBA bus interface logic
120d
120c
Rd Lock Fail Q
Order Q
Read Q
Hi Priority Queue
120a
120b
FBox address and command queues
head tail
3
read requests Order Q Hi Priority
Insert queue control
remove Q arbitration
127
F0, F1, F2, F3, F4, and F5 data pull_wr/push_rd engine
131
service priority machine
Mem Amba Data
32
128
Core Memory Write Data
Core Memory Requests
AMBA Address Rd/Wr Queue 1
126
138
amd decoder and address generator
AM_en
sr_en
rd tail
140
pad_data
pad_en
SRAM Interface Logic
Insert/ Remove Queue SM
addr
142
8 entry address CAM for locks
A
B
C
D
data[31:0]
address
rd/wr/en
16b
SRAM 32KB to 8MB pipeline/ flwthru 12-15ns
16c
BootRom upto 6MB
to console port on MAC chips

FIG. 5-1

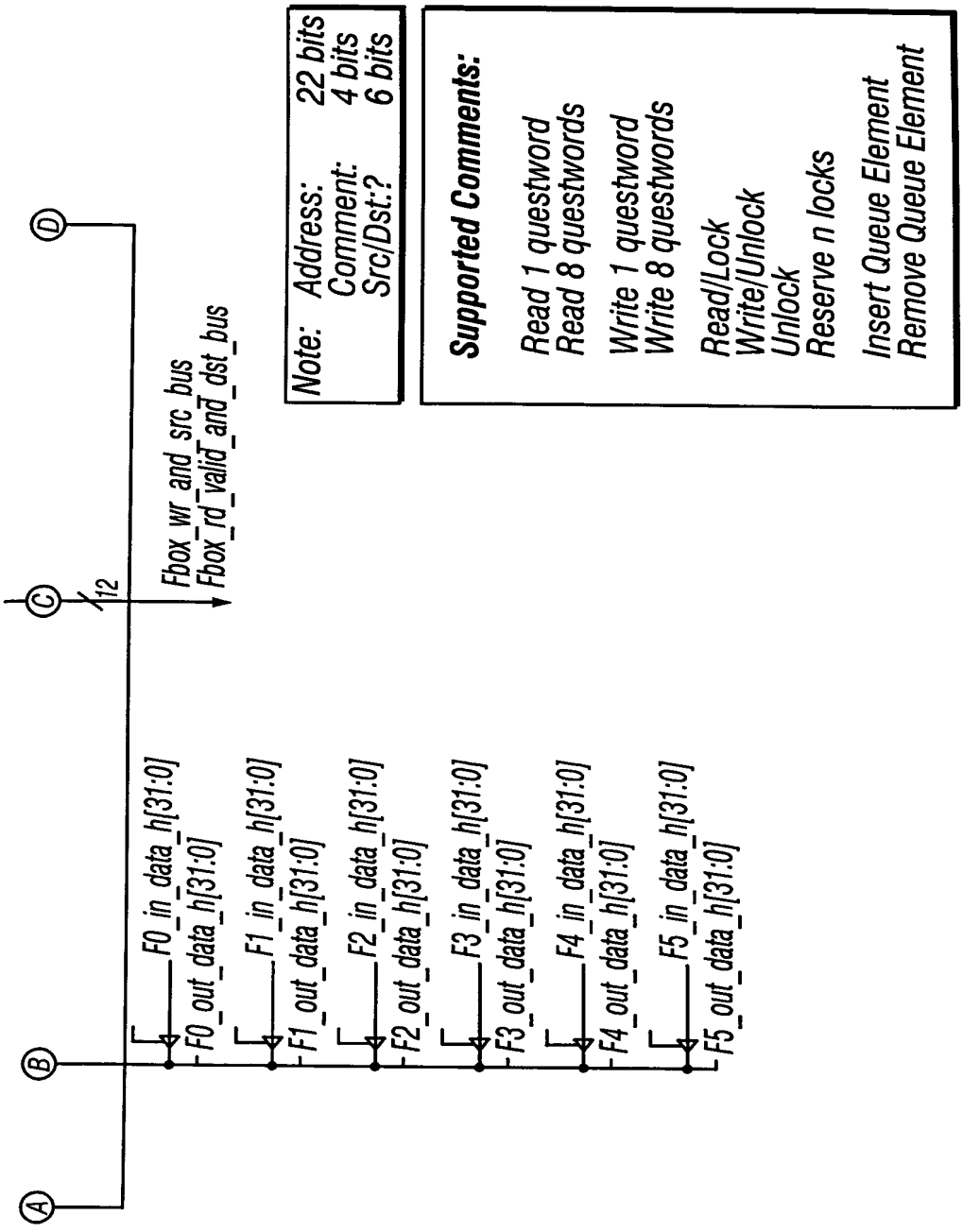

A
B
C
D
12
Fbox_wr_and_src_bus
Fbox_rd_valid_and_dst_bus
F0_in_data_h[31:0]
F0_out_data_h[31:0]
F1_in_data_h[31:0]
F1_out_data_h[31:0]
F2_in_data_h[31:0]
F2_out_data_h[31:0]
F3_in_data_h[31:0]
F3_out_data_h[31:0]
F4_in_data_h[31:0]
F4_out_data_h[31:0]
F5_in_data_h[31:0]
F5_out_data_h[31:0]
Note: Address: 22 bits
Comment: 4 bits
Src/Dst:? 6 bits
Supported Comments:
Read 1 questword
Read 8 questwords
Write 1 questword
Write 8 questwords
Read/Lock
Write/Unlock
Unlock
Reserve n locks
Insert Queue Element
Remove Queue Element

FIG. 5-2

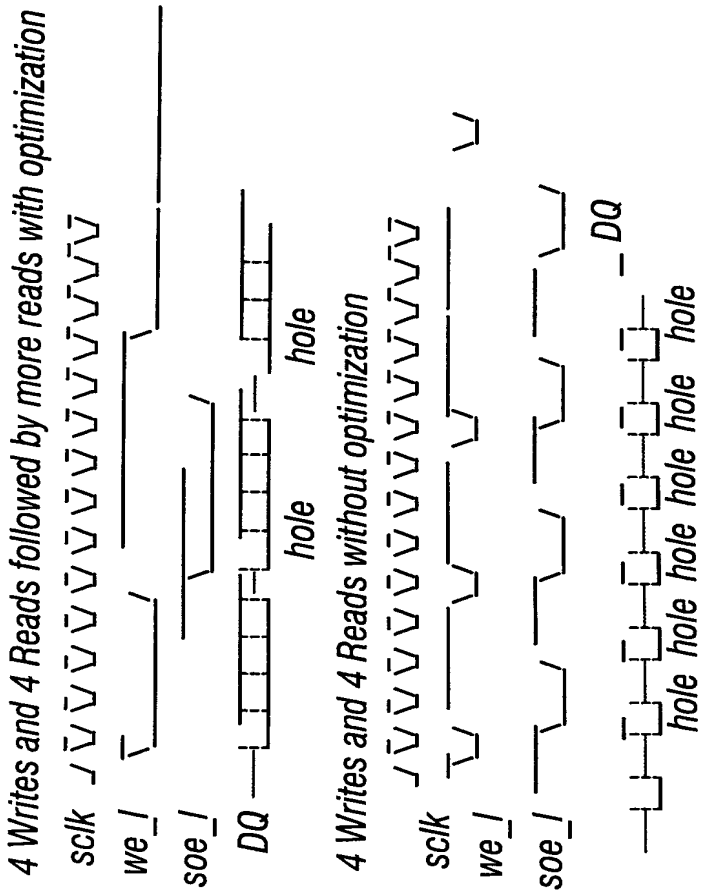

4 Writes and 4 Reads followed by more reads with optimization
sclk
we_l
soe_l
DQ
hole
hole
4 Writes and 4 Reads without optimization
sclk
we_l
soe_l
DQ
hole hole hole hole hole hole hole hole
10 cycles vs. 14

FIG. 5A

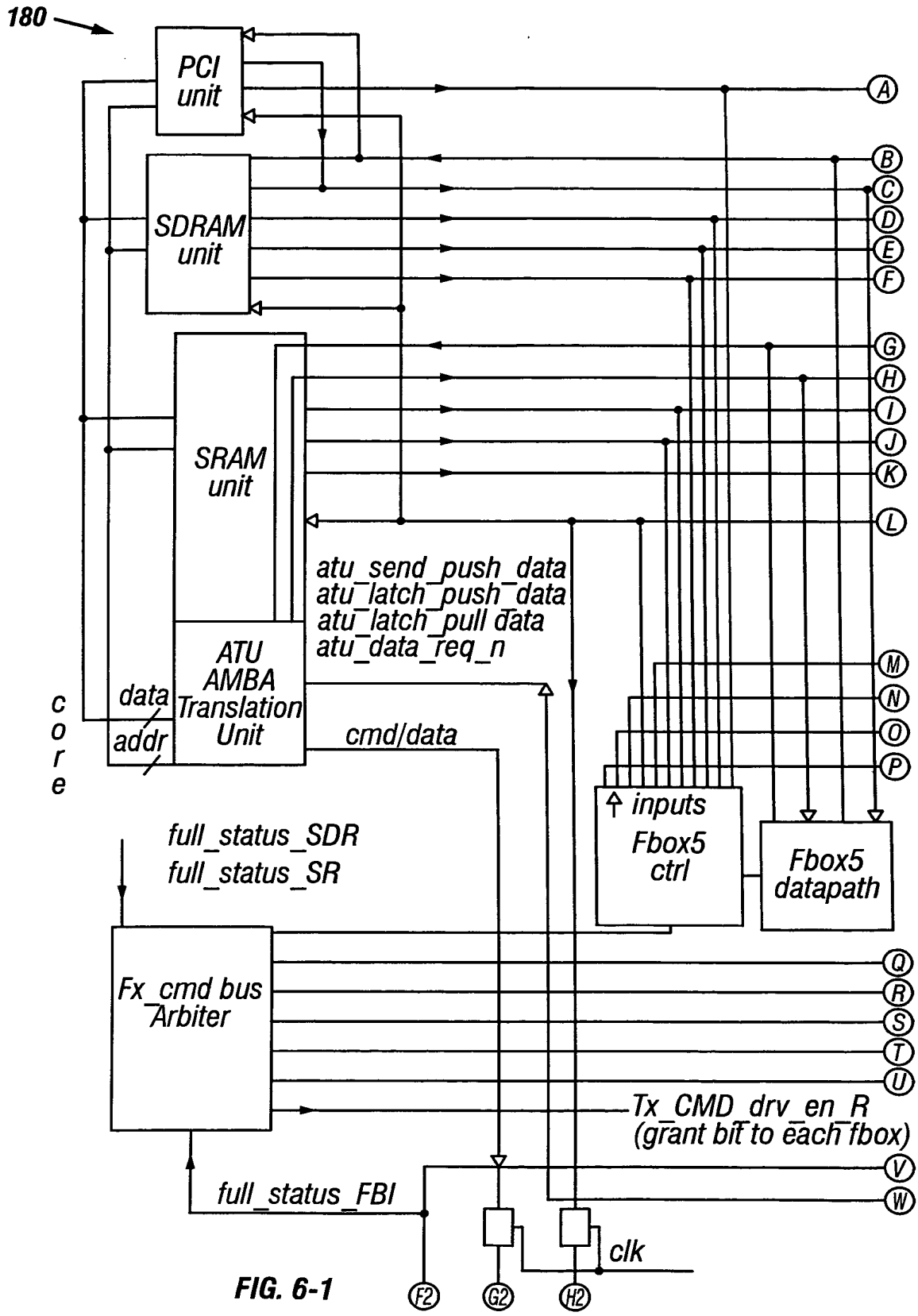

180
PCI unit
SDRAM unit
SRAM unit
ATU AMBA Translation Unit
data
addr
core
atu_send_push_data
atu_latch_push_data
atu_latch_pull data
atu_data_req_n
cmd/data
inputs
Fbox5 ctrl
Fbox5 datapath
full_status_SDR
full_status_SR
Fx_cmd bus Arbiter
Tx_CMD_drv_en_R
(grant bit to each fbox)
full_status_FBI
clk
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
F2
G2
H2

FIG. 6-1

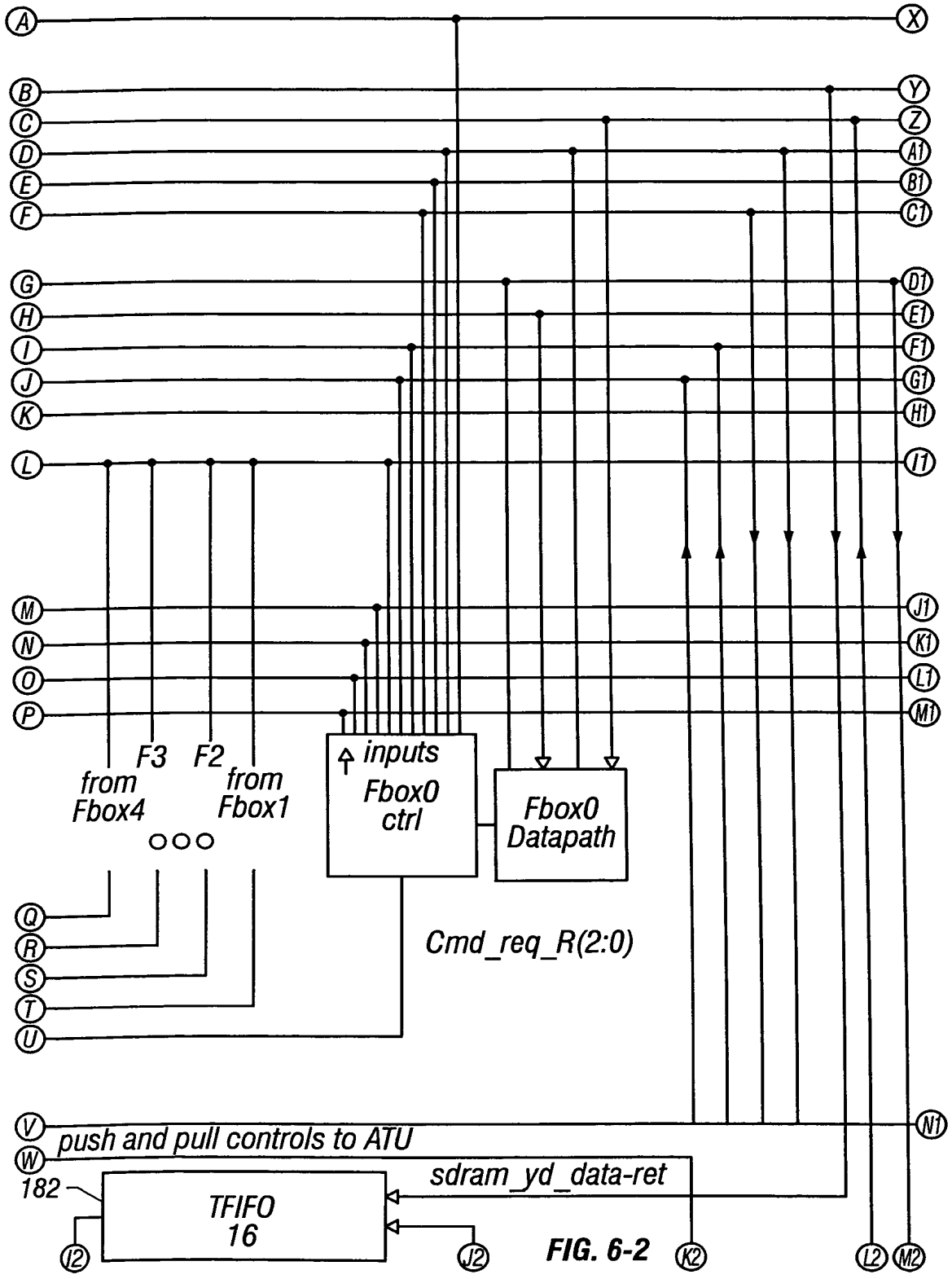

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
A1
B1
C1
D1
E1
F1
G1
H1
I1
J1
K1
L1
M1
N1
F3
F2
from Fbox4
from Fbox1
inputs
Fbox0 ctrl
Fbox0 Datapath
Cmd_req_R(2:0)
push and pull controls to ATU
182
TFIFO 16
sdram_yd_data-ret
I2
J2
K2
L2
M2

FIG. 6-2

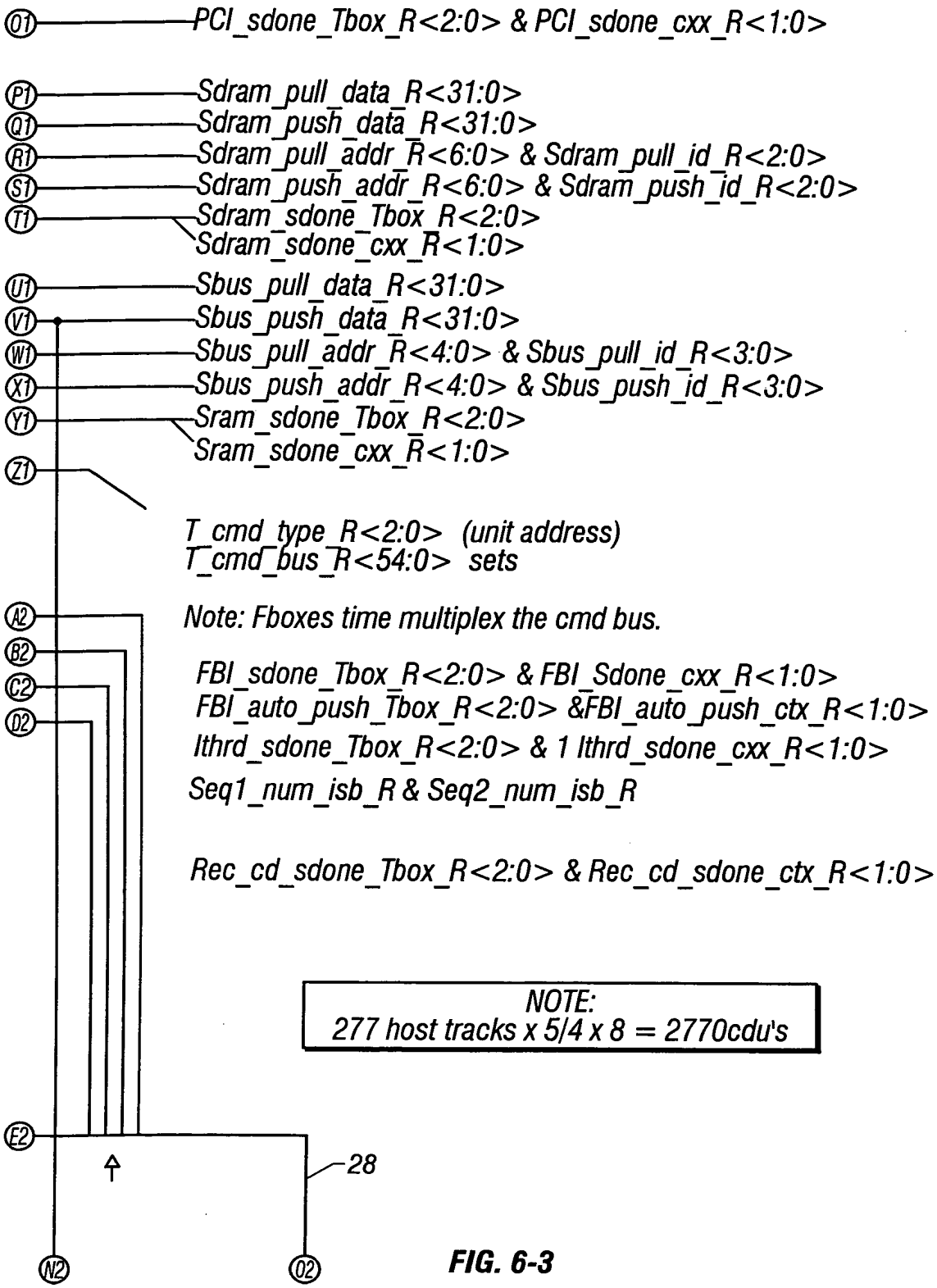

O1
PCI_sdone_Tbox_R<2:0> & PCI_sdone_cxx_R<1:0>
P1
Sdram_pull_data_R<31:0>
Q1
Sdram_push_data_R<31:0>
R1
Sdram_pull_addr_R<6:0> & Sdram_pull_id_R<2:0>
S1
Sdram_push_addr_R<6:0> & Sdram_push_id_R<2:0>
T1
Sdram_sdone_Tbox_R<2:0>
Sdram_sdone_cxx_R<1:0>
U1
Sbus_pull_data_R<31:0>
V1
Sbus_push_data_R<31:0>
W1
Sbus_pull_addr_R<4:0> & Sbus_pull_id_R<3:0>
X1
Sbus_push_addr_R<4:0> & Sbus_push_id_R<3:0>
Y1
Sram_sdone_Tbox_R<2:0>
Sram_sdone_cxx_R<1:0>
Z1
T_cmd_type_R<2:0> (unit address)
T_cmd_bus_R<54:0> sets
Note: Fboxes time multiplex the cmd bus.
A2
B2
C2
D2
FBI_sdone_Tbox_R<2:0> & FBI_Sdone_cxx_R<1:0>
FBI_auto_push_Tbox_R<2:0> &FBI_auto_push_ctx_R<1:0>
Ithrd_sdone_Tbox_R<2:0> & 1 Ithrd_sdone_cxx_R<1:0>
Seq1_num_isb_R & Seq2_num_isb_R
Rec_cd_sdone_Tbox_R<2:0> & Rec_cd_sdone_ctx_R<1:0>
NOTE:
277 host tracks x 5/4 x 8 = 2770cdu's
E2
28
N2
O2

FIG. 6-3

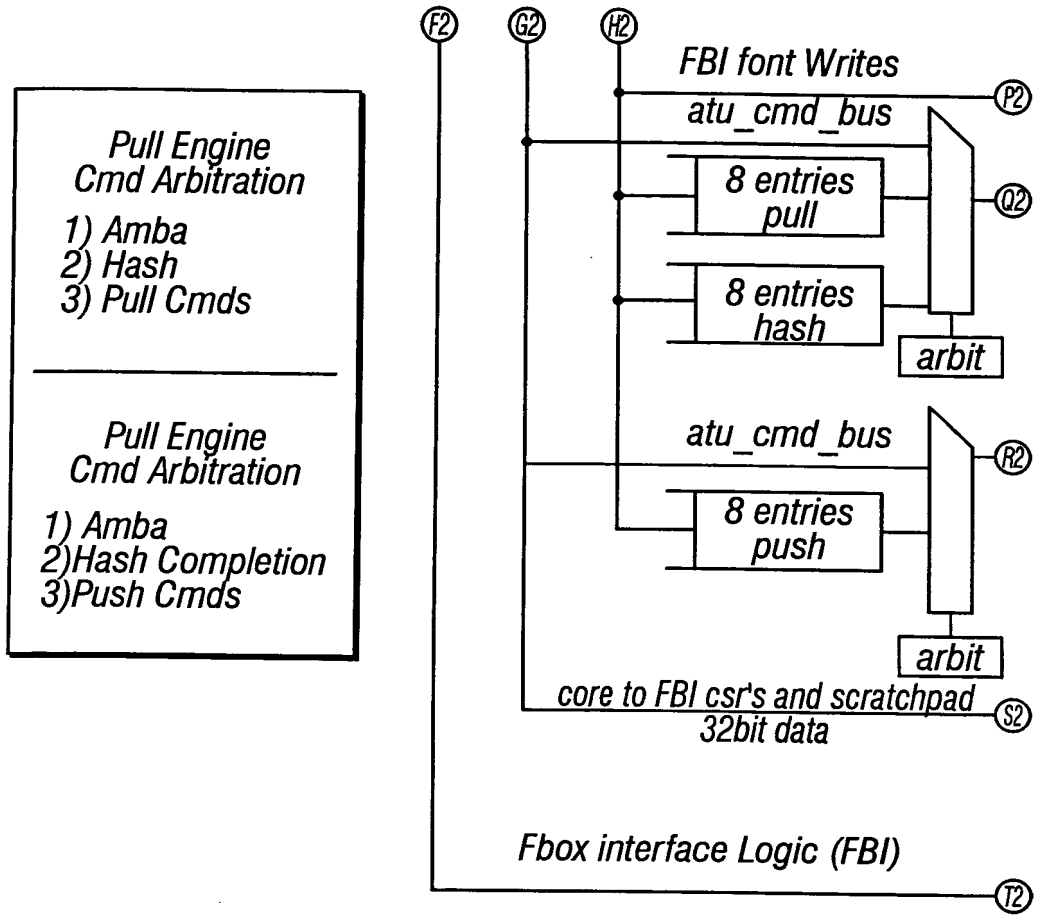

F2
G2
H2
FBI font Writes
atu_cmd_bus
P2
8 entries pull
Q2
8 entries hash
arbit
atu_cmd_bus
R2
8 entries push
arbit
core to FBI csr's and scratchpad
32bit data
S2
Fbox interface Logic (FBI)
T2
Pull Engine Cmd Arbitration
1) Amba
2) Hash
3) Pull Cmds
Pull Engine Cmd Arbitration
1) Amba
2)Hash Completion
3)Push Cmds
ATU Notes:
a) Core to FboxRegs: use sram_push_data_bus
b) Core to FBI Regs: use private ATU/FBI cmd/data bus
c) Core reads FboxRegs: use SRAM_pull_data_bus
d) Core reads FBIRegs: use sram_push_data_bus (makes sram appear like another Fbox to FBI on sram_push_bus)
Cmd_Req_R<2:0>
000 none
001 Sram Chain
010 SDR chain
011 Sram
100 SDR
101 FBI
110 PCI
111
Tx_CMD_drv_en_R<1:0>
0 none
1 grant

FIG. 6-4

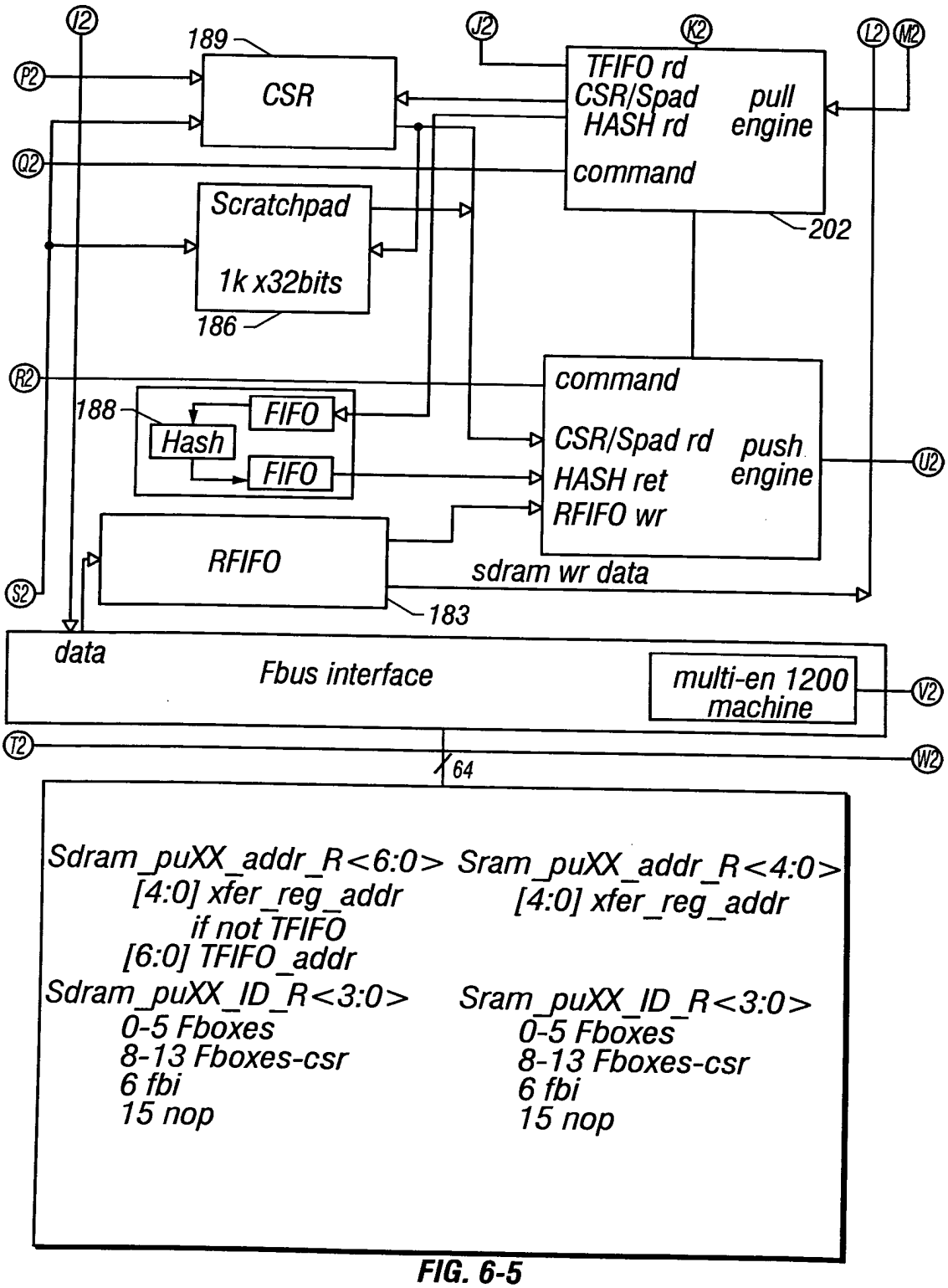

I2
189
J2
K2
L2
M2
P2
CSR
TFIFO rd
CSR/Spad
HASH rd
pull
engine
Q2
command
Scratchpad
1k x32bits
202
186
R2
command
188
FIFO
Hash
FIFO
CSR/Spad rd
HASH ret
RFIFO wr
push
engine
U2
RFIFO
sdram wr data
S2
183
data
Fbus interface
multi-en 1200
machine
V2
T2
W2
64
Sdram_puXX_addr_R<6:0>
[4:0] xfer_reg_addr
if not TFIFO
[6:0] TFIFO_addr
Sram_puXX_addr_R<4:0>
[4:0] xfer_reg_addr
Sdram_puXX_ID_R<3:0>
0-5 Fboxes
8-13 Fboxes-csr
6 fbi
15 nop
Sram_puXX_ID_R<3:0>
0-5 Fboxes
8-13 Fboxes-csr
6 fbi
15 nop

FIG. 6-5

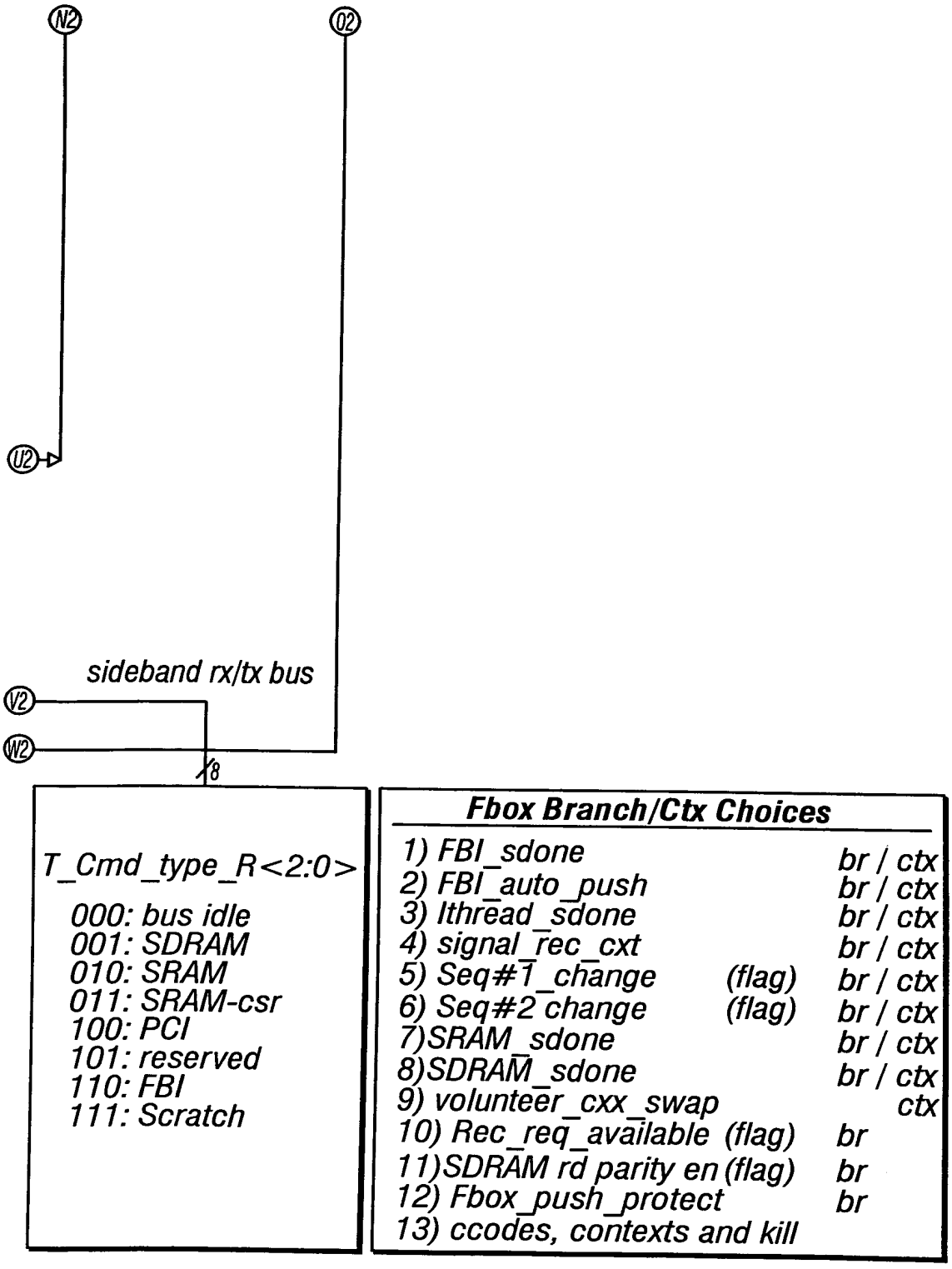

N2
O2
U2
sideband rx/tx bus
V2
W2
8
T_Cmd_type_R<2:0>
000: bus idle
001: SDRAM
010: SRAM
011: SRAM-csr
100: PCI
101: reserved
110: FBI
111: Scratch
Fbox Branch/Ctx Choices
1) FBI_sdone br / ctx
2) FBI_auto_push br / ctx
3) Ithread_sdone br / ctx
4) signal_rec_cxt br / ctx
5) Seq#1_change (flag) br / ctx
6) Seq#2 change (flag) br / ctx
7)SRAM_sdone br / ctx
8)SDRAM_sdone br / ctx
9) volunteer_cxx_swap ctx
10) Rec_req_available (flag) br
11)SDRAM rd parity en (flag) br
12) Fbox_push_protect br
13) ccodes, contexts and kill

FIG. 6-6